

Contents

1 Routine/Function Prologues	2
1.1 Fortran: Module Interface opendap_module.F90 (Source File: opendap_module.F90)	2
1.1.1 opendap_init (Source File: opendap_module.F90)	3
1.1.2 opendap_readcard (Source File: opendap_module.F90)	3
1.1.3 reset_lis_filepaths (Source File: opendap_module.F90)	3
1.1.4 init_parm_vars (Source File: opendap_module.F90)	4
1.1.5 set_parm_lat (Source File: opendap_module.F90)	6

1 Routine/Function Prologues

1.1 Fortran: Module Interface opendap_module.F90 (Source File: opendap_module.F90)

This module contains routines needed to initialize and control variables required for the execution of GDS-based I/O

REVISION HISTORY:

10 Jul 2003; James Geiger Initial Specification

INTERFACE:

```
module opendap_module
#if ( defined OPENDAP )
```

USES:

```
use ESMF_TimeMgmtMod
use time_manager
use lisdrv_module, only : lis, grid, gindex
use grid_spmdMod,   only : gdi, gdisp
use tile_spmdMod,   only : di_array
use spmdMod

implicit none
```

ARGUMENTS:

integer	:: tnroffset	!Global to local row mapping offset
integer	:: grid_offset	!Global to local grid mapping offset
integer	:: output_slat	Southern latitude boundary for the interpolated domain
integer	:: output_nlat	Northern latitude boundary for the interpolated domain
integer	:: output_wlon	Western longitude boundary for the interpolated domain
integer	:: output_elon	Eastern longitude boundary for the interpolated domain
integer	:: parm_slat	Southern latitude boundary for each subdomain (for parameter data)
integer	:: parm_nlat	Northern latitude boundary for each subdomain (for parameter data)
integer	:: parm_wlon	Western longitude boundary for each subdomain (for parameter data)
integer	:: parm_elon	Eastern longitude boundary for each subdomain (for parameter data)
integer	:: parm_nc	Number of columns for each subdomain (for parameter data)
integer	:: parm_nr	Number of rows for each subdomain (for parameter data)
character*3	:: ciam	character representation of processor id
character*5	:: cparm_slat	character representation of cparm\$-\$slat
character*5	:: cparm_nlat	character representation of cparm\$-\$nlat
character*5	:: cparm_wlon	character representation of cparm\$-\$wlon
character*5	:: cparm_elon	character representation of cparm\$-\$elon
character*40	:: opendap_data_prefix	
integer	:: nroffset	

1.1.1 opendap_init (Source File: *opendap_module.F90*)

Initializes the GDS variables

INTERFACE:

```
subroutine opendap_init()
```

CONTENTS:

```
call init_parm_vars()
call reset_lis_filepaths()
```

1.1.2 opendap_readcard (Source File: *opendap_module.F90*)

Reads the opendap namelist from the lis.crd card file

INTERFACE:

```
subroutine opendap_readcard()
```

CONTENTS:

```
open(10,file='lis.crd',form='formatted',status='old')
read(unit=10,nml=opendap)
print*, 'MSG: opendap_readcard -- Using data prefix ', &
        opendap_data_prefix, ' (', iam, ')'
close(10)
```

1.1.3 reset_lis_filepaths (Source File: *opendap_module.F90*)

Resets various input data filenames for execution through GDS

INTERFACE:

```
subroutine reset_lis_filepaths()
```

CONTENTS:

```
lis%p%clfile      = trim(opendap_data_prefix)//'/'// &
                     trim(adjustl(ciam))//'/'//lis%p%clfile
lis%p%safile      = trim(opendap_data_prefix)//'/'// &
                     trim(adjustl(ciam))//'/'//lis%p%safile
lis%p%iscfile     = trim(opendap_data_prefix)//'/'// &
                     trim(adjustl(ciam))//'/'//lis%p%iscfile
```

1.1.4 init_parm_vars (Source File: opendap_module.F90)

Computes domain decomposition for native as well as interpolated domains for input data

INTERFACE:

```
subroutine init_parm_vars()
```

CONTENTS:

```
select case (lis%d%domain)
case(1)
    print*, 'Error!  Cannot handle nldas.'
    stop 999
case(2) ! 1 deg
    res = 1000
    center = 500
    s_origin = -59500
    w_origin = -179500
case(3)
    print*, 'Error!  Cannot handle 2x2.5.'
    stop 999
case(4) ! 1/4 deg
    res = 250
    center = 875
    s_origin = -59875
    w_origin = -179875
case(5) ! 1/2 deg
    res = 500
    center = 750
    s_origin = -59750
    w_origin = -179750
case(6) ! 5km
    res = 50
    center = 975
    s_origin = -59975
    w_origin = -179975
case(7) ! 5km
    res = 50
    center = 975
    s_origin = -59975
    w_origin = -179975
case(8) ! 1km
    res = 10
    center = 995
    s_origin = -59995
    w_origin = -179995
case DEFAULT
    print*, "Select domain size (1,2,3,4,5,6,7,8)"
    stop 999
```

```

end select

call set_parm_lat(parm_slat,parm_nlat,output_slat,output_nlat,&
                  parm_wlon,parm_elon,output_wlon,output_elon,&
                  res,s_origin,w_origin)
!parm_wlon = 1
!parm_elon = lis%d%gnc
!parm_wlon = 1 + lis%d%gnc * (lis%d%ic - 1)
!parm_elon = lis%d%gnc + lis%d%gnc * (lis%d%ic - 1)

lis%d%ngrid = gdi(iam)
lis%d%nch    = di_array(iam)

!parm_nc      = lis%d%gnc
parm_nc      = ( parm_elon - parm_wlon + 1 )
parm_nr      = ( parm_nlat - parm_slat + 1 )

if ( lis%d%ir > 1 ) then ! running the special 1km regional domain
    tnroffset = parm_slat - 1 - (lis%d%ir - 1)*lis%d%gnr
else ! running the ‘‘normal’’ way
    tnroffset = parm_slat - 1
endif
nroffset = 0
grid_offset = tile(1)%index-1

write(ciam, '(i3)') iam
write(cparm_slat, '(i5)') parm_slat
write(cparm_nlat, '(i5)') parm_nlat
write(cparm_wlon, '(i5)') parm_wlon
write(cparm_elon, '(i5)') parm_elon

print*, 'DBG: parm_init -- parm_slat', parm_slat, ', (', iam, ')'
print*, 'DBG: parm_init -- parm_nlat', parm_nlat, ', (', iam, ')'
print*, 'DBG: parm_init -- parm_wlon', parm_wlon, ', (', iam, ')'
print*, 'DBG: parm_init -- parm_elon', parm_elon, ', (', iam, ')'
print*, 'DBG: parm_init -- output_slat', output_slat, ', (', iam, ')'
print*, 'DBG: parm_init -- output_nlat', output_nlat, ', (', iam, ')'
print*, 'DBG: parm_init -- output_wlon', output_wlon, ', (', iam, ')'
print*, 'DBG: parm_init -- output_elon', output_elon, ', (', iam, ')'
print*, 'DBG: parm_init -- ngrid', lis%d%ngrid, ', (', iam, ')'
print*, 'DBG: parm_init -- glbngrid', lis%d%glbngrid, ', (', iam, ')'
print*, 'DBG: parm_init -- lnc', lis%d%lnc, ', (', iam, ')'
print*, 'DBG: parm_init -- lnr', lis%d%lnr, ', (', iam, ')'
print*, 'DBG: parm_init -- gnc', lis%d%gnc, ', (', iam, ')'
print*, 'DBG: parm_init -- gnr', lis%d%gnr, ', (', iam, ')'
print*, 'DBG: parm_init -- tnroffset', tnroffset, ', (', iam, ')'
print*, 'DBG: parm_init -- grid_offset', grid_offset, ', (', iam, ')'
print*, 'DBG: parm_init -- nch', lis%d%nch, ', (', iam, ')'

```

```

print*,'DBG: parm_init -- ngrid', lis%d%ngrid, '(, iam, )'
print*,'DBG: parm_init -- glbnch', lis%d%glbnch, '(, iam, )'
print*,'DBG: parm_init -- glbngrid', lis%d%glbngrid, '(, iam, )'
print*,'DBG: parm_init -- cparm_slat ', cparm_slat, '(, iam, )'
print*,'DBG: parm_init -- cparm_nlat ', cparm_nlat, '(, iam, )'
print*,'DBG: parm_init -- ciam ', ciam, '(, iam, )'
print*,'DBG: parm_init -- parm_nc', parm_nc, '(, iam, )'
print*,'DBG: parm_init -- parm_nr', parm_nr, '(, iam, )'
print*,'DBG: parm_init -- gdi(iam)', gdi(iam), '(, iam, )'
print*,'DBG: parm_init -- tile(1)%row', tile(1)%row, '(, iam, )'
print*,'DBG: parm_init -- tile(1)%col', tile(1)%col, '(, iam, )'
print*,'DBG: parm_init -- tile(gdi(iam))%row', tile(gdi(iam))%row, &
      '(, iam, )'
print*,'DBG: parm_init -- tile(gdi(iam))%col', tile(gdi(iam))%col, &
      '(, iam, )'

```

1.1.5 set_parm_lat (Source File: opendap_module.F90)

Computes the latitudes of the decomposed domain for the parameter data

INTERFACE:

```

subroutine set_parm_lat(slat, nlat, oslat, onlat, &
                       wlon, elon, owlon, oelon, &
                       res, s_origin, w_origin)

```

USES:

```

use lisdrv_module, only : tile
implicit none

```

INPUT PARAMETERS:

```

integer, intent(in) :: res, s_origin, w_origin

```

OUTPUT PARAMETERS:

```

integer, intent(out) :: slat, nlat, oslat, onlat, &
                      wlon, elon, owlon, oelon

```

CONTENTS:

```

! Set southern latitude index
!slat = tile( 1 )%row
slat = tile( 1 )%row + (lis%d%ir-1)*lis%d%gnr

! Set southern latitude boundary
oslat = s_origin + (slat-1)*res

```

```
! Set northern latitude index
!nlat = tile( gdi(iam) )%row
nlat = tile( gdi(iam) )%row + (lis%d%ir-1)*lis%d%gnr

! Set northern latitude boundary
onlat = s_origin + (nlat-1)*res

! Set western longitude index
wlon = 1 + lis%d%gnc * (lis%d%ic - 1)

! Set western longitude boundary
owlon = w_origin + (wlon-1)*res

! Set eastern longitude index
elon = lis%d%gnc + lis%d%gnc * (lis%d%ic - 1)

! Set eastern longitude boundary
oelon = w_origin + (elon-1)*res

end subroutine set_parm_lat
#endif
```